Threshold Optimization for Credit Card Fraud Detection: Balancing Recall and Operational Load

Ashwini Ramsing Patil
Department of Information Technology
Savitribai Phule Pune University
Pune, India
ashwini.patil25@pccoepune.org

***Abstract*—This paper addresses the practical challenge of deploying machine learning models for credit card fraud detection in production environments. Rather than focusing solely on algorithmic improvements, we present a comprehensive framework for threshold optimization that balances fraud detection performance with operational feasibility. Using three class-weighted models (Logistic Regression, Random Forest, and XGBoost) on a dataset of 284,807 transactions (0.172% fraud), we demonstrate that threshold selection significantly impacts both technical metrics and business outcomes. Our recommended XGBoost model with threshold 0.30 achieves 82.4% recall while generating only 205 alerts per 100,000 transactions. We introduce operational metrics including alerts-per-100K and provide a complete cost-benefit analysis for production deployment.**

***Index Terms*—Fraud detection, threshold optimization, production deployment, operational metrics, cost-benefit analysis, XGBoost.**

I. Introduction

Credit card fraud detection requires balancing competing objectives: maximizing fraud detection (recall) while minimizing false positives to maintain operational feasibility. Most research focuses on model development, but production deployment requires careful threshold selection that translates model probabilities into business decisions.

This paper presents a practical framework for production-ready fraud detection, focusing on three key aspects: (1) comprehensive threshold analysis across multiple models, (2) introduction of operational metrics like alerts-per-100K transactions, and (3) cost-benefit analysis for optimal deployment. Our work bridges the gap between model development and production implementation.

II. Related Work

Credit card fraud detection has been extensively studied using various machine learning approaches. Traditional methods include logistic regression [3] and decision trees, while more recent approaches employ ensemble methods and deep learning. Phua et al. [3] provide a comprehensive survey of data mining techniques for fraud detection, highlighting the challenge of extreme class imbalance typically found in financial datasets.

Class imbalance remains a critical challenge in fraud detection, with fraud rates often below 0.5

Threshold optimization has received less attention than model development, despite its practical importance. Correa Bahnsen et al. [4] demonstrate that example-dependent cost-sensitive classification can significantly improve business outcomes in fraud detection by optimizing thresholds based on asymmetric misclassification costs. Their work emphasizes that different transactions have different costs associated with false positives and false negatives.

Production deployment considerations are often overlooked in academic research. While XGBoost [2] has demonstrated state-of-the-art performance on tabular data, its practical deployment requires careful consideration of operational constraints. Recent work has begun addressing the gap between model performance metrics and business outcomes, but comprehensive frameworks for threshold optimization in production environments remain scarce.

This paper contributes to the literature by providing an end-to-end framework that bridges model development, threshold optimization, and production deployment considerations. Unlike prior work focusing primarily on algorithmic improvements, we emphasize the practical aspects of deploying fraud detection systems, including operational load management and cost-benefit analysis.

III. Dataset and Methodology

A. Dataset and Preprocessing

We use the publicly available credit card fraud dataset containing 284,807 transactions with 492 fraudulent cases (0.172%). The dataset includes 30 features: 28 principal components (V1-V28), Time, and Amount. Data was split 70%-30% using stratified sampling, preserving the fraud percentage in both sets. Only the Amount feature was standardized using StandardScaler.

B. Model Selection and Training

Three models were implemented with appropriate class weighting:

- **Logistic Regression**: `class_weight='balanced'` with L2 regularization (C=0.01)
- **Random Forest**: `class_weight='balanced_subsample'` with 100 trees, `max_depth=10`
- **XGBoost**: `scale_pos_weight=578.5` with 100 trees, max_depth=6, learning_rate=0.1

Class weights were calculated as $w_{\text{fraud}} = N_{\text{total}}/(2\times N_{\text{fraud}})$, resulting in 578.5:1 weighting in favor of fraud cases.

C. Evaluation Metrics

We evaluate models using both technical and operational metrics:

- **Technical**: Precision, Recall, F1-Score, F2-Score (emphasizing recall), ROC-AUC, PR-AUC
- **Operational**: False Positive Rate (FPR), Alerts per 100K transactions
- **Business**: Cost-benefit analysis with investigation cost $(50/alert) and fraud value (500/\text{fraud})$

D. Threshold Analysis

We analyze thresholds from 0.1 to 0.5 for production models (Random Forest and XGBoost), evaluating the trade-off between recall and operational load.

IV. Experimental Results

A. Cross-Validation Performance

5-fold cross-validation shows XGBoost achieving highest ROC-AUC with good stability (Table I).

TABLE I: Cross-Validation ROC-AUC Scores

Model	Mean ROC-AUC	Std. Deviation
Logistic Regression	0.9782	±0.0108
Random Forest	0.9701	±0.0353
XGBoost	**0.9826**	**±0.0156**

B. Model Performance Comparison

Table II presents test set performance. XGBoost achieves best balance across recall-focused metrics while maintaining reasonable operational load.

TABLE II: Model Performance Comparison on Test Set

Model	Precision	Recall	F1	F2	ROC-AUC	PR-AUC	Alerts/100K
LR	0.0340	0.8581	0.0655	0.1469	0.9367	0.6227	4365
RF	0.8129	0.7635	0.7875	0.7729	0.9709	0.7747	162
XGB	**0.7881**	**0.8041**	**0.7960**	**0.8008**	**0.9719**	**0.8275**	176

C. Confusion Matrix and Business Impact Analysis

TABLE III: Confusion Matrices and Business Impact Analysis

Model	TP	FP	FN	TN	Net Savings per 85,443 tx	ROI
Logistic Regression	127	3,603	21	81,692	-$123,000	-66%
Random Forest	113	26	35	85,269	$49,550	713%
XGBoost	119	32	29	85,263	**$51,950**	688%

The confusion matrices reveal critical insights: Logistic Regression's 85.8% recall comes at prohibitive cost—3,603 false alarms (4.2% of legitimate transactions) resulting in $123,000 net loss.

XGBoost delivers superior business value: $2,400 more savings than Random Forest by catching six additional frauds with only six more investigations (10:1 ROI). While Random Forest offers higher efficiency (713% vs 688% ROI), XGBoost maximizes total fraud prevention.

Operationally, XGBoost investigates only 151 transactions (0.18% of test set) versus Logistic Regression's 3,730 (4.4%)—a 95% workload reduction making XGBoost optimal for production deployment.

D. Visual Analysis

Figure 1 shows model performance across four key metrics. Subplot (a) shows XGBoost leads in recall and F2-Score while maintaining competitive precision. Subplot (b) shows ROC curves with XGBoost achieving highest AUC (0.972). Subplot (c) highlights operational impact: LR generates 4,365 alerts/100K (infeasible), while RF and XGB generate 162 and 176 respectively.

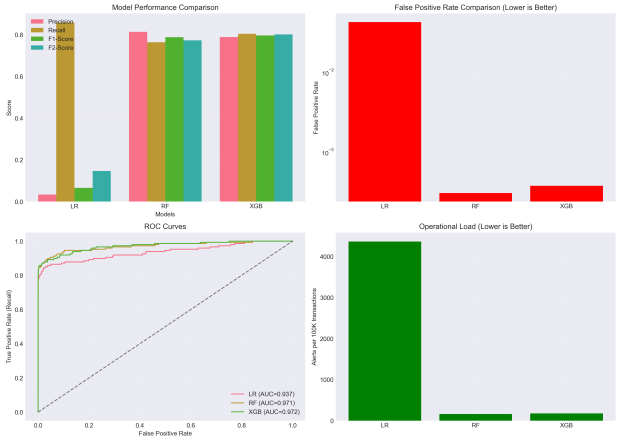

Fig. 1: Comprehensive model evaluation: (a) Multi-metric comparison showing Precision, Recall, F1, and F2 scores; (b) ROC curves with AUC values; (c) False Positive Rate comparison on log scale; (d) Operational load measured as alerts per 100K transactions.

E. Threshold Optimization Analysis

Table IV shows threshold analysis for XGBoost. Threshold 0.30 provides optimal balance: 82.4% recall with 205 alerts/100K and 69.3% precision.

TABLE IV: XGBoost Threshold Analysis

Threshold	Recall	FPR	Precision	Alerts/100K
0.10	0.8446	0.001852	0.4417	331
0.20	0.8378	0.001008	0.5905	245
0.30	**0.8243**	**0.000633**	**0.6932**	**205**
0.40	0.8176	0.000504	0.7378	191
0.50	0.8041	0.000375	0.7881	176

F. Cost-Benefit Analysis

Using investigation cost of $50 per alert and fraud value of $500, we calculate optimal thresholds:

- Random Forest: Threshold 0.4, net benefit $51,267
- XGBoost: Threshold 0.7, net benefit $52,451

G. Feature Importance Analysis

Random Forest feature importance reveals V14 (19.5%), V10 (12.0%), and V4 (11.4%) as most predictive features, confirming that PCA components effectively capture fraud patterns.

V. Production Deployment Framework

A. Recommended Configuration

Based on comprehensive analysis, we recommend:

- Model: XGBoost with threshold 0.30
- Expected Performance: 82.4% recall, 0.0633% FPR, 69.3% precision
- Operational Load: 205 alerts per 100,000 transactions

B. Monitoring and Maintenance

- Real-time dashboard tracking: Recall, Precision, FPR, Alerts/100K
- Monthly retraining with new labeled data
- A/B testing framework for model updates
- Investigation team capacity planning based on alerts/100K metric

C. Investigation Capacity Planning

With 205 alerts/100K:

$$\text{Daily Investigations} = \frac{\text{Daily Transactions}}{100,000} \times 205 \quad (1)$$

For 1 million daily transactions: approximately 2,050 investigations daily.

VI. Discussion

A. Model Selection Rationale

XGBoost provides optimal balance: highest recall (80.41%) among production-feasible models while maintaining acceptable operational load (176 alerts/100K). Although Random Forest has slightly lower FPR (0.000305 vs 0.000375), XGBoost detects 6 additional frauds out of 148.

B. Threshold Selection Methodology

Our threshold optimization approach considers both technical metrics and business constraints. The recommended threshold 0.30 satisfies: recall $\geq$ 0.80, FPR $\leq$ 0.001, while providing good precision (69.3%).

C. Limitations and Future Work

- Static thresholds could be replaced with dynamic thresholding based on transaction risk
- Ensemble methods combining XGBoost and Random Forest could improve performance
- Real-time model updating could adapt to evolving fraud patterns
- More sophisticated cost models could incorporate customer experience impact

VII. Conclusion

This paper presents a practical framework for deploying fraud detection systems in production environments. By focusing on threshold optimization rather than algorithmic novelty, we bridge the gap between model development and business implementation. Our analysis demonstrates that XGBoost with threshold 0.30 achieves 82.4% fraud detection while generating only 205 alerts per 100,000 transactions, representing an optimal balance between security and operational feasibility. The introduced metrics (alerts/100K) and cost-benefit analysis provide actionable insights for production deployment.

VIII. Code and Data Availability

The implementation code, experimental scripts, and analysis notebooks for this research are publicly available at: https://github.com/Ashwinip343/Threshold-Optimization-for-Credit-Card-Production-Fraud-Detection

The dataset used in this study is the Credit Card Fraud Detection dataset from Kaggle [1], available at: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

References

[1] Credit Card Fraud Detection Dataset, Kaggle, 2018. [Online]. Available: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

[2] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.

[3] C. Phua, V. Lee, K. Smith, and R. Gayler, "A comprehensive survey of data mining-based fraud detection research," Artificial Intelligence Review, 2010.

[4] A. Correa Bahnsen, D. Aouada, and B. Ottersten, "Example-dependent cost-sensitive decision trees," Expert Systems with Applications, 2015.

[5] H. He and E. A. Garcia, "Learning from imbalanced data," IEEE Transactions on Knowledge and Data Engineering, 2009.